

Algoritmos e Lógica de Programação

80 horas // 4 h/semana

***Expressões
matemáticas e
lógicas. Tipo de
dados***

Aula 03

Prof. Piva

Expressões matemáticas...

- As expressões matemáticas (e lógicas) como conhecemos e utilizamos, não podem ser implementadas no computador no formato que conhecemos.
- Elas devem sofrer um processo conhecido como **linearização**. Além disso, existe um conjunto de operações matemáticas que deve ser implementado para possibilitar que instruções gráficas, como raiz quadrada, possam ser devidamente utilizadas. A maioria dessas operações será implementada como funções.

Linearização

- Uma expressão matemática convencional:

$$x = \frac{3y}{5y + 7} + 2y$$

- A mesma expressão linearizada:

$$x \leftarrow ((3 * y) / (5 * y + 7)) + (2 * y)$$

Operadores matemáticos...

TABELA 3.1: Principais operadores matemáticos em linguagem algorítmica

Operações	Operador	Exemplo
Adição	+	$a + b$ (a mais b)
Subtração	-	$a - b$ (a menos b)
Multiplicação	*	$a * b$ (a vezes b)
Divisão	/	a / b (a dividido por b)

Linearização...

Vamos ver mais um exemplo para fixar:

$$y = \frac{x + 3b}{2x + c}$$

Essa mesma expressão linearizada, ficaria da seguinte forma:

$$y \leftarrow (x + 3 * b) / (2 * x + c)$$

Mais operadores matemáticos...

TABELA 3.2: Operadores e funções para operações matemáticas mais complexas em linguagem algorítmica

Operações	Operador	Exemplo
Exponenciação	\wedge	$a \wedge b$ (a elevado a b)
Divisão Inteira	\backslash	$a \backslash b$ (valor inteiro resultante da divisão de a por b)
Módulo (resto da divisão)	$\%$	$a \% b$ (resto da divisão de a por b)
Inversão de Sinal	$-$	$a \text{ / } b$ ($-(-a)$ resulta em a)
Operações	Funções	Explicação
Raiz quadrada	Raizq(x)	Raiz quadrada de x.
Exponenciação	Exp(x,y)	x elevado a y.
Valor absoluto (sem sinal)	Abs(x)	Valor absoluto de x.
Arco Co-seno	ArcCos(x)	Retorna o ângulo (em radianos) cujo cosseno é representado por x.

Mais linearização....

(1)

$$y = \frac{a^2 + \sqrt{3b}}{5x^3}$$

(2)

$$x = y + \sqrt{\frac{2b}{a+b}}$$

Linearizando a primeira expressão, ela ficaria da seguinte forma:

$$y \leftarrow (\text{quad}(a) + \text{raizq}(3 * b)) / (5 * \text{exp}(b, 3))$$

Linearizando a segunda expressão, ela ficaria da seguinte forma:

$$X \leftarrow y + \text{raizq}((2 * b) / (a + b))$$

Operadores relacionais...

TABELA 3.3: Operadores relacionais em linguagem algorítmica

Operações	Operador	Exemplo
Igual	=	$a = b$ (a é igual a b?)
diferente	\neq	$a \neq b$ (a é diferente de b?)
Maior que	$>$	$a > b$ (a é maior que b?)
Menor que	$<$	$a < b$ (a é menor que b?)
Maior ou igual que	\geq	$a \geq b$ (a é maior ou igual a b?)
Menor ou igual que	\leq	$a \leq b$ (a é menor ou igual a b?)

Operadores lógicos...

TABELA 3.4: Operadores lógicos em linguagem algorítmica

Operadores	Significado
nao	Operador unário de negação. Tem a maior precedência entre os operadores lógicos. nao VERDADEIRO = FALSO, e nao FALSO = VERDADEIRO.
ou	Operador que resulta VERDADEIRO quando um dos seus operandos lógicos for verdadeiro.
e	Operador que resulta VERDADEIRO somente se seus dois operandos lógicos forem verdadeiros.
xou	Operador que resulta VERDADEIRO se seus dois operandos lógicos forem diferentes, e FALSO se forem iguais.

Operadores lógicos...

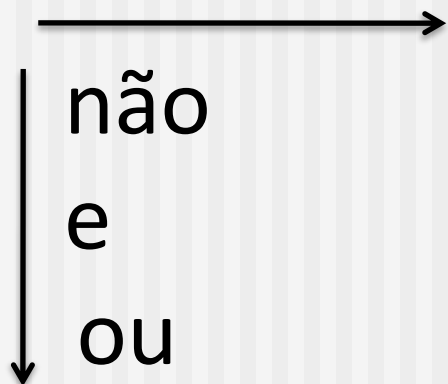
TABELA 3.6: Tabela Verdade do operador “E”

A	B	S = A e B
Falso	Falso	Falso
Falso	Verdadeiro	Falso
Verdadeiro	Falso	Falso
Verdadeiro	Verdadeiro	Verdadeiro

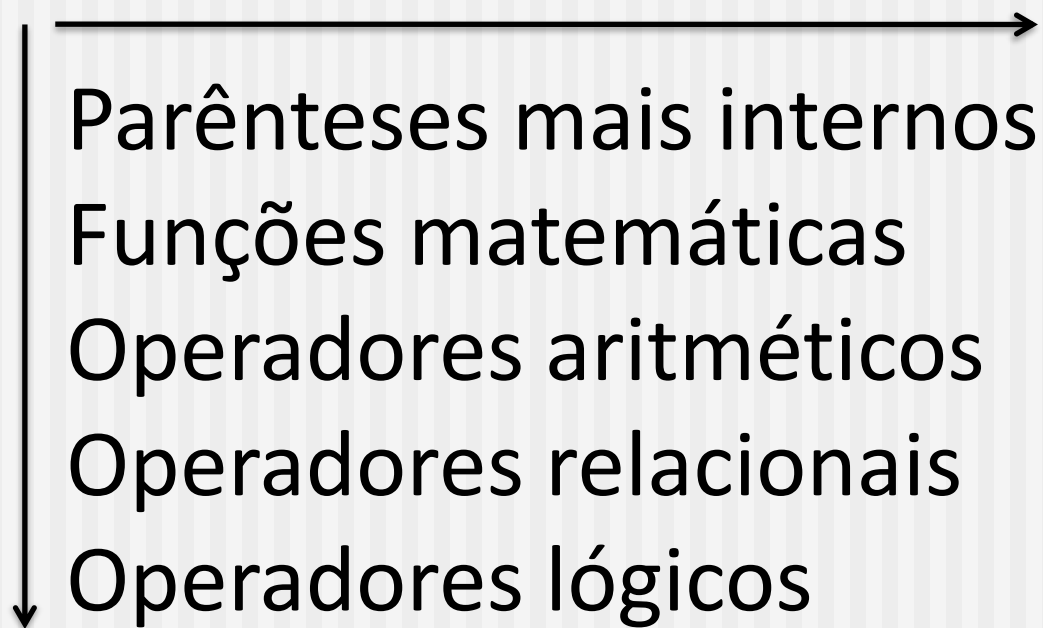
TABELA 3.7: Tabela Verdade do operador “OU”

A	B	S = A ou B
Falso	Falso	Falso
Falso	Verdadeiro	Verdadeiro
Verdadeiro	Falso	Verdadeiro
Verdadeiro	Verdadeiro	Verdadeiro

Prioridades – Operadores Lógicos



Prioridades – Todos os Operadores...



Agora é com você...

1) Indique o resultado das seguintes expressões:

a) $2 > 3$

b) $(6 < 8) \text{ OR } (3 > 7)$

c) $(((10 \text{ DIV } 2) \text{ MOD } 6) > 5) \text{ XOR } (3 < (2 \text{ MOD } 2))$

d) $\text{NOT } (2 < 3)$

Agora é com você...

2) Escreva o comando de atribuição e resolva a expressão das seguintes expressões matemáticas (implemente o comando de atribuição em todas as linguagens vistas: VisuAlg, Pascal, C, Java e PHP).

$$\text{a) } X = \frac{A + \frac{B}{C}}{D - \frac{E}{F}} \text{ onde } A=2, B=6, C=3, D=4, E=8, F=4$$

$$\text{b) } Y = \frac{\frac{2X^2 - 3X^{(X+1)}}{2} + \frac{\sqrt{X+1}}{X}}{2^X} \text{ onde } X=2$$

Agora é com você...

3) Construa o algoritmo que calcule as seguintes expressões:

a) $2 + 3 * \{ 23 - 12 + [\{ (123 / 34) + 10 \} / 2 - 1 * (45 - 12)] / 3 \}$

b) $[(2 * 3) ^2 - 1] ^4$

c) $(2 + [2 * 3 - (4 / 20) ^2]) / (10 * \{ [(7 * 3) - 2] ^3 \})$

Agora é com você...

4) Escreva os comandos de atribuição (em todas as linguagens vistas) para as seguintes expressões matemáticas linearização.

a)
$$X = \frac{\sqrt{2B - 4A} + 2F^{-3}}{3 - 2A}$$

b)
$$Y = 2H - \left[\frac{45}{3X} - 4H(3 - H) \right]^{2H}$$

TIPO DE DADOS

Tipo de dados...

- Pensando em termos de dados utilizados nos programas, ou seja, valores, letras, nomes que servirão de “entrada” ou “matéria-prima” para realizarmos as tarefas/ações planejadas em nosso algoritmo/programa, deveremos levar em consideração o tamanho e as características de cada um.
- Dependendo de seu valor ou variedade desse dado, uma quantidade maior ou menor de memória devera ser alocada para guarda-lo.
- A essa quantidade de memória reservada previamente, e devidamente rotulada com um nome, chamamos de **variável**.

Tipos Básicos...

- Inteiro
 - 654, -4567, 89, 3700, 0, -1
- Real
 - 876.90 , 1.5 , 36548987.99876
- Caracter
 - 0..9 / A..Z / a..z
 - Caracteres especiais: #, /, %, *, ? ...
- Lógico
 - Verdadeiro ou Falso

Variáveis...

Regras para nomes de identificadores

- a) devem começar com um caractere alfabético;
- b) podem ser seguidas por mais caracteres alfabéticos e/ou numéricos;
- c) é permitido o uso do caractere especial 'sublinha' (_);
- d) não é permitido o uso de outros caracteres especiais.

Exemplos de identificadores

Não permitidos	1ABC, EF*GH, DT ANIVER
Permitidos	X, NOME, A12, SAL1
Aconselháveis	nome, salarioLiquido, variavel1

Variáveis...

Regras para nomes de identificadores

■ Padrões de Projetos:

Pelos padrões de projeto (*design patterns*), as variáveis devem ser escritas em letras minúsculas. Caso exista a necessidade de expressá-las em 2 ou mais palavras, estas, a partir da 2ª devem ter sua primeira letra escrita em maiúscula. Exemplos: dataDeNascimento ou salarioBruto.

Agora é com você...

Valores	VISUALG
127	
"Z"	
Falso	
0.556789075	
-456	
"Algoritmo"	

EXERCÍCIOS

EXERCÍCIO 1

(MANZANO; OLIVEIRA, 2000 - pág. 35) Indique com um X quais dos dados abaixo são do tipo **Inteiro**.

☐ 1000

☐ "0"

☐ "-900"

☐ .Verdadeiro.

☐ -456

☐ 34

☐ "Casa 8"

☐ 0

☐ .Falso.

☐ -1.56

EXERCÍCIO 2

(MANZANO; OLIVEIRA, 2000 - pág. 35) Indique com um X quais dos dados abaixo são do tipo **Real**.

- ☐ -6.78
- ☐ "0.87"
- ☐ "-9.12"
- ☐ .Verdadeiro.
- ☐ -456
- ☐ -99.8
- ☐ "Cinco"
- ☐ 45.8976
- ☐ .Falso.
- ☐ -1.56

EXERCÍCIO 3

(MANZANO; OLIVEIRA, 2000 - pág. 35) Indique com um X quais dos dados abaixo são do tipo **Literal (Caractere)**.

- ☐ 678
- ☐ "0.87"
- ☐ "-9.12"
- ☐ "Verdadeiro"
- ☐ -456
- ☐ -99.8
- ☐ "Cinco"
- ☐ 45.8976
- ☐ .Falso.
- ☐ 1.56

EXERCÍCIO 4

(MANZANO; OLIVEIRA, 2000 - pág. 36) Indique com um X quais dos dados abaixo são do tipo **Lógico**.

- () -678
- () "0.87"
- () "-9.12"
- () .Verdadeiro.
- () -456
- () .V.
- () "Cinco"
- () .Falso.
- () .F.
- () -1.56

EXERCÍCIO 5

(MANZANO; OLIVEIRA, 2000 - pág. 36) : Assinale com X os nomes válidos para uma variável.

- () ENDERECO
- () 21 BRASIL
- () FONE\$COM
- () NOMEUSUÁRIO
- () NOME USUÁRIO
- () NOME*USUÁRIO
- () END*A-6
- () CIDADE3
- () #CABEC

EXERCÍCIO 6

(FORBELLONE; EBERSPÄCHER, 2000 - pág. 18) Assinale os identificadores válidos:

- | | |
|--------------------------------------|-------------------------------------|
| a) <input type="checkbox"/> (X) | i) <input type="checkbox"/> AB*C |
| b) <input type="checkbox"/> U2 | j) <input type="checkbox"/> O&O |
| c) <input type="checkbox"/> AH! | l) <input type="checkbox"/> P{0} |
| d) <input type="checkbox"/> "ALUNO" | m) <input type="checkbox"/> B52 |
| e) <input type="checkbox"/> #55 | n) <input type="checkbox"/> Rua |
| f) <input type="checkbox"/> KM/L | o) <input type="checkbox"/> CEP |
| g) <input type="checkbox"/> UYT | p) <input type="checkbox"/> dia/mês |
| h) <input type="checkbox"/> ASDRUBAL | |

EXERCÍCIO 7

(FORBELLONE; EBERSPÄCHER, 2000 - pág. 18) Supondo que as variáveis NB, NA, Nmat, SX sejam utilizadas para armazenar a nota do aluno, o nome do aluno, o número da matrícula e o sexo, declare-as corretamente, associando o tipo primitivo adequado ao dado que será armazenado.

EXERCÍCIO 8

(FORBELLONE; EBERSPÄCHER, 2000 - pág. 18) Encontre os erros da seguinte declaração de variáveis:

inteiro: Endereço, NFilhos;

caracter: Idade, X;

real: XPTO, C, Peso, R\$;

lógico: Lâmpada, C;

EXERCÍCIO 9

(FORBELLONE; EBERSPÄCHER, 2000 - pág. 25) Determine os resultados obtidos na avaliação das expressões lógicas seguintes, sabendo que A, B, C contêm, respectivamente, 2, 7, 3,5, e que existe uma variável lógica L cujo valor é falsidade (F).

- a)** $B = A * C \text{ e } (L \text{ ou } V)$
- b)** $B > A \text{ ou } B = \text{pot}(A, A)$
- c)** $L \text{ e } B \text{ div } A \geq C \text{ ou não } A \leq C$
- d)** $\text{não } L \text{ ou } V \text{ e } \text{rad}(A + B) \geq C$
- e)** $B / A = C \text{ ou } B / A \neq C$
- f)** $L \text{ ou } \text{pot}(B, A) \leq C * 10 + A * B$

VAMOS PARA A PRÁTICA ?!!!



Python: Tipos Básicos

Python como calculadora

- O Interpretador python pode ser usado como calculadora
- Por exemplo, as quatro operações aritméticas são denotadas pelos símbolos
 - + adição
 - - subtração
 - * multiplicação
 - / divisão

Python como calculadora

```
>>> 10
10
>>> # Um comentário é precedido do caractere "#"
... # Comentários são ignorados pelo interpretador
... 10+5
15
>>> 10-15 # Comentários podem aparecer também após código
-5
>>> 10*3
30
>>> 10/3
3
>>> 10/-3 # Divisão inteira retorna o piso
-4
>>> 10%3 # Resto de divisão inteira simbolizado por %
1
```

Tipos de dados

- São categorias de valores que são processados de forma semelhante
- Por exemplo, números inteiros são processados de forma diferente dos números de ponto flutuante (decimais) e dos números complexos
- Tipos primitivos: são aqueles já embutidos no núcleo da linguagem
 - Simples: números (int, long, float, complex) e cadeias de caracteres (strings)
 - Compostos: listas, dicionários, tuplas e conjuntos
- Tipos definidos pelo usuário: são correspondentes a classes (orientação objeto)

Variáveis

- São nomes dados a áreas de memória
 - Nomes podem ser compostos de algarismos, letras ou _
 - O primeiro caractere não pode ser um algarismo
 - Palavras reservadas (if, while, etc) são proibidas
- Servem para:
 - Guardar valores intermediários
 - Construir estruturas de dados
- Uma variável é modificada usando o comando de atribuição:
Var = expressão
- É possível também atribuir a várias variáveis simultaneamente:
var1,var2,...,varN = expr1,expr2,...,exprN

Variáveis

```
>>> a=1
```

```
>>> a
```

```
1
```

```
>>> a=2*a
```

```
>>> a
```

```
2
```

```
>>> a,b=3*a,a
```

```
>>> a,b
```

```
(6,2)
```

```
>>> a,b=b,a
```

```
>>> a,b
```

```
(2,6)
```

Variáveis

- Variáveis são criadas dinamicamente e destruídas quando não mais necessárias, por exemplo, quando saem fora de escopo (veremos isso mais tarde)
- O *tipo* de uma variável muda conforme o valor atribuído, i.e., int, float, string, etc.

- Não confundir com linguagens *sem tipo*

- Ex.:

```
>>> a = "1"
```

```
>>> b = 1
```

```
>>> a+b
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in ?
```

```
TypeError: cannot concatenate 'str' and 'int' objects
```


Números

- Há vários tipos numéricos que se pode usar em python
 - **Int:** números inteiros de *precisão fixa*
 - 1 , 2 , 15 , -19
 - **Long:** números inteiros de *precisão arbitrária*
 - 1L , 10000L , -9999999L
 - **Floats:** números racionais de *precisão variável*
 - 1.0 , 10.5 , -19000.00005 , 15e-5
 - **Complex:** números complexos
 - 1+1j , 20j , 1000+100J

Números inteiros

- Os **ints** têm precisão fixa ocupando tipicamente uma palavra de memória
 - Em PC's são tipicamente representados com 32 bits (de $-2^{31}-1$ a 2^{32})
- Os números inteiros de precisão arbitrária (**longs**) são armazenados em tantas palavras quanto necessário
 - Constantes do tipo long têm o sufixo L ou l
 - Longs são manipulados bem mais lentamente que ints
 - Quando necessário, cálculos usando ints são convertidos para longs

Números inteiros

```
>>> a=2**30 # Potenciação
```

```
>>> a
```

```
1073741824
```

```
>>> b=a*1000
```

```
>>> b
```

```
1073741824000L
```

```
>>> b/1000
```

```
1073741824L
```

Números inteiros

- Constantes podem ser escritas com notação idêntica à usada em C

- Hexadecimal: preceder dígitos de 0x

- Octal: preceder dígitos de 0

- Ex.:

```
>>> 022
```

```
18
```

```
>>> 0x10
```

```
16
```

```
>>> 0x1f
```

```
31
```

Números de ponto flutuante

- São implementados como os **double**'s da linguagem C – tipicamente usam 2 palavras
- Constantes têm que possuir um ponto decimal ou serem escritas em notação científica com a letra “e” (ou “E”) precedendo a potência de 10
- Ex:
 - >>> 10 # inteiro
10
 - >>> 10.0 # ponto flutuante
10.0
 - >>> 99e3
99000.0
 - >>> 99e-3
0.099000000000000005

Números complexos

- Representados com dois números de ponto flutuante: um para a parte real e outro para a parte imaginária
- Constantes são escritas como uma soma sendo que a parte imaginária tem o sufixo j ou J
- Ex.:
 - >>> 1+2j
(1+2j)
 - >>> 1+2j*3
(1+6j)
 - >>> (1+2j)*3
(3+6j)
 - >>> (1+2j)*3j
(-6+3j)

Strings

- São cadeias de caracteres
- Constituem outro tipo fundamental do python
- Constantes ***string*** são escritas usando aspas simples ou duplas
 - Ex.: "a" ou 'a'
- O operador "+" pode ser usado para concatenar strings
 - Ex.: "a"+"b" é o mesmo que "ab"
- O operador "*" pode ser usado para repetir strings
 - Ex.: "a"*10 é o mesmo que "aaaaaaaaaa"

Strings

- Python usa a tabela de caracteres default do S.O.
 - Ex.: ASCII, UTF-8
- Caracteres não imprimíveis podem ser expressos usando notação “barra-invertida” (\)
 - \n é o mesmo que *new line*
 - \r é o mesmo que *carriage return*
 - \t é o mesmo que *tab*
 - \b é o mesmo que *backspace*
 - \\ é o mesmo que \
 - \x41 é o mesmo que o caractere cujo código hexadecimal é 41 (“A” maiúsculo)

Strings

```
>>> "ab\r d"
```

```
'ab\r d'
```

```
>>> print ("ab\r d") # print exibe chars não imprimíveis
```

```
ab
```

```
>>> print ("abc\td")
```

```
abc    d
```

```
>>> print ("abc\nd")
```

```
abc
```

```
d
```

```
>>> print ("abc\\nd")
```

```
abc\nd
```

```
>>> print ("ab\bc")
```

```
ac
```

```
>>> print ("\x41\xA1")
```

```
Aí
```

Strings

- A notação *barra-invertida* (\) pode ser desabilitada desde que a constante string seja precedida por um r (erre minúsculo)

- São chamadas strings *raw* (cruas)

- Ex.:

```
>>> print ("abc\n cd\tef")
```

```
abc
```

```
cd    ef
```

```
>>> print (r"abc\n cd\tef")
```

```
abc\n cd\tef
```

Strings

- Constantes string podem ser escritas com várias linhas desde que as aspas não sejam fechadas e que cada linha termine com uma barra invertida

- Ex.:

```
>>> print ("abcd\n\  
... efgh\n\  
... ijk"  
abcd  
efgh  
ijk  
>>> print ("abcd\  
... efgh\  
... ijk"  
abcdefghijk  
>>>
```

Strings

- Também é possível escrever constantes string em várias linhas incluindo as quebras de linha usando três aspas como delimitadores

- Ex.:

```
>>> print ("""  
Um tigre  
dois tigres  
três tigres""")
```

```
Um tigre  
dois tigres  
três tigres  
>>> print ("abcd  
efgh")  
abcd  
efgh
```

Strings – Índices

- Endereçam caracteres individuais de uma string

- Notação: *string[índice]*
- O primeiro caractere tem índice 0
- O último caractere tem índice -1

- Ex.:

```
>>> a = "abcde"
```

```
>>> a[0]
```

```
'a'
```

```
>>> a[-1]
```

```
'e'
```

Strings – Fatias (slices)

- Notação para separar trechos de uma string
 - Notação: *string[índice1:índice2]*
 - Retorna os caracteres desde o de índice1 (inclusive) até o de índice2 (exclusive)
 - Se o primeiro índice é omitido, é assumido 0
 - Se o último índice é omitido, é assumido o fim da string

Strings – Fatias (slices)

```
>>> a
'abcde'
>>> a[0:2]
'ab'
>>> a [2:]
'cde'
>>> a[:]
'abcde'
>>> a[-1:]
'e'
>>> a[:-1]
'abcd'
```

Expressões booleanas

- Também chamadas expressões lógicas
- Resultam em verdadeiro (True) ou falso (False)
- São usadas em comandos condicionais e de repetição
- Servem para analisar o estado de uma computação e permitir escolher o próximo passo
- Operadores mais usados
 - Relacionais: $>$, $<$, $==$, $!=$, $>=$, $<=$
 - Booleanos: and, or, not
- Avaliação feita em “Curto-circuito”
 - Expressão avaliada da esquerda para a direita
 - Se o resultado (verdadeiro ou falso) puder ser determinado sem avaliar o restante, este é retornado imediatamente

Expressões booleanas

```
>>> 1==1
```

```
True
```

```
>>> 1==2
```

```
False
```

```
>>> 1==1 or 1==2
```

```
True
```

```
>>> 1==1 and 1==2
```

```
False
```

```
>>> 1<2 and 2<3
```

```
True
```

```
>>> not 1<2
```

```
False
```

```
>>> not 1<2 or 2<3
```

```
True
```

```
>>> not (1<2 or 2<3)
```

```
False
```

```
>>> "alo" and 1
```

```
1
```

```
>>> "alo" or 1
```

```
'alo'
```

Expressões booleanas

- As constantes **True** e **False** são apenas símbolos convenientes
- Qualquer valor não nulo é visto como verdadeiro enquanto que **0** (ou **False**) é visto como falso
- O operador **or** retorna o primeiro operando se for vista como *verdadeiro*, caso contrário retorna o segundo
- O operador **and** retorna o primeiro operando se for vista como *falso*, caso contrário retorna o segundo
- Operadores relacionais são avaliados antes de **not**, que é avaliado antes de **and**, que é avaliado antes de **or**

Expressões booleanas

```
>>> 0 or 100
```

```
100
```

```
>>> False or 100
```

```
100
```

```
>>> "abc" or 1
```

```
'abc'
```

```
>>> 1 and 2
```

```
2
```

```
>>> 0 and 3
```

```
0
```

```
>>> False and 3
```

```
False
```

```
>>> 1 and 2 or 3
```

```
2
```

```
>>> 0 or 2 and 3
```

```
3
```

```
>>> 1 and not 0
```

```
True
```

Funções Embutidas

- Além dos operadores, é possível usar funções para computar valores
- As funções podem ser definidas:
 - Pelo programador (veremos + tarde)
 - Em módulos da biblioteca padrão
 - Por *default*: são as funções *embutidas* (*built-in*)
 - Na verdade, fazem parte do módulo `__builtins__`, que é sempre importado em toda aplicação
- Ex.:
 - `abs(x)` retorna o valor absoluto do número `x`
 - `chr(x)` retorna uma string com um único caractere cujo código ASCII é `x`
 - `ord(s)` retorna o código ASCII do caractere `s`

Funções Embutidas

```
>>> abs (10)
```

```
10
```

```
>>> abs (-19)
```

```
19
```

```
>>> chr (95)
```

```
'_'
```

```
>>> chr (99)
```

```
'c'
```

```
>>> ord ('a')
```

```
97
```

Importando módulos

- Muitas funções importantes são disponibilizadas em módulos da biblioteca padrão
 - Ex.: o módulo **math** tem funções transcendentais como **sin**, **cos**, **exp** e outras
- Um módulo pode conter não só funções mas também variáveis ou classes
 - Por exemplo, o módulo **math** define a constante **pi**
- Para usar os elementos de um módulo, pode-se usar o comando **import**
 - Formatos:
 - **import** *modulo*
 - **from** modulo **import** *nome,...,nome*
 - **from** modulo **import** *

Importando módulos

■ Por exemplo:

- `from math import *`
importa todos os elementos do módulo `math`
- `from math import sin`
importa apenas a função `sin`
- `import math`
importa o módulo `math` como um todo
(todos os elementos têm que ser citados
precedidos por **`math.`**)

Importando módulos

```
>>> import math
```

```
>>> a = sin(30)
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in ?
```

```
NameError: name 'sin' is not defined
```

```
>>> a = math.sin(30)
```

```
>>> from math import sin
```

```
>>> a = sin(30)
```

```
>>> print (a)
```

```
-0.988031624093
```

```
>>> a = sin(radians(30))
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in ?
```

```
NameError: name 'radians' is not defined
```

```
>>> from math import *
```

```
>>> a = sin(radians(30))
```

```
>>> a
```

```
0.49999999999999994
```


Explorando Módulos

```
>>> import math
```

```
>>> help(math.cos)
```

Help on built-in function cos in module math:

```
cos(...)
```

```
    cos(x)
```

Return the cosine of x (measured in radians).

```
(END)
```

■ Pressiona-se “q” para retornar ao interpretador.